MANUAL DE USUARIO COMPLETO

Sistema de Gestión del Árbol Genealógico - Civilización Antigua

Implementación en C++ y Módulo Complementario en Python

INTRODUCCIÓN

El presente manual describe el funcionamiento integral del Sistema de Gestión del Árbol Genealógico, compuesto por dos componentes principales:

1. **Programa principal en C++**: Sistema completo de gestión de miembros de un árbol genealógico con estructura de datos de árbol binario de búsqueda.
2. **Módulo complementario en Python**: Herramienta de visualización gráfica y balanceo automático del árbol.

Ambos sistemas operan de manera independiente pero se complementan para proporcionar una solución completa de gestión, análisis y optimización de estructuras de árbol binario de búsqueda.

El objetivo es brindar una explicación clara, formal y sencilla del funcionamiento interno de ambos sistemas, sus validaciones, limitaciones y recomendaciones para un uso eficiente.

PARTE I: SISTEMA PRINCIPAL EN C++

1. ESTRUCTURA PRINCIPAL DEL SISTEMA

El programa utiliza una estructura dinámica basada en nodos conectados mediante punteros. Cada nodo representa un miembro del árbol genealógico.

Estructura Nodo:

- id (int)
- nombre (string)
- anio (int)
- genero (char)
- rol (string)
- izq (Nodo*) → hijo izquierdo
- der (Nodo*) → hijo derecho
- activo (bool) → indica si el nodo está disponible o eliminado lógicamente

La raíz del árbol es un puntero de tipo Nodo* llamado "raiz", inicializado en NULL.

2. OBJETIVO DEL PROGRAMA EN C++

El programa tiene como finalidad:

- Administrar miembros mediante una estructura arbórea ordenada por su ID.
- Permitir operaciones de consulta y análisis de la forma y contenido del árbol.
- Facilitar que el usuario explore conceptos de estructuras de datos, ABB y algoritmos recursivos.
- Proveer una herramienta educativa para cursos de programación y estructuras de datos.

3. FUNCIONAMIENTO GENERAL DEL SISTEMA C++

Al ejecutarse, el programa:

1. Inicializa un puntero de 100 posiciones mediante la función iniArreglo(), asignando nodos vacíos.
2. Muestra un menú principal con cinco secciones:
   - Gestión de Miembros
   - Recorridos
   - Información del Árbol
   - Operaciones Avanzadas
   - Salir

Cada sección agrupa funciones específicas que operan sobre el árbol.

El programa opera de manera iterativa hasta que el usuario selecciona la opción "Salir".

4. ESTRUCTURA DEL ÁRBOL Y DEL NODO

Cada nodo del árbol contiene los siguientes campos:

- **ID**: identificador único del miembro
- **Nombre**
- **Año de nacimiento**
- **Género** (M/F)
- **Rol social**
- **Estado** (1 = activo, -1 = inactivo/eliminado)

Estos nodos se organizan según un árbol binario de búsqueda, comparando únicamente el ID:

- Si el nuevo ID es menor que el del nodo actual → va a la izquierda

- Si es mayor → va a la derecha

Cada hijo se ubica en la posición del puntero:

- Hijo izquierdo = 2 × índice
- Hijo derecho = 2 × índice + 1

5. MÓDULOS DEL SISTEMA C++

5.1 Gestión de Miembros

a) Insertar miembro

Función: insertar()

- Solicita al usuario todos los datos del miembro
- Valida:
  - ID positivo
  - ID no existente (consulta secuencial)
  - Género correcto
  - Año positivo
- Inserta el nodo realizando comparaciones de ID hasta hallar su posición vacía

Si el puntero está lleno o la posición índice supera 99, muestra mensaje de error.

b) Buscar miembro

Función: buscar()

- Recorre todo el puntero y muestra los IDs válidos
- Solicita ingresar el ID buscado
- Si lo encuentra, se muestran todos sus datos
- Si no existe, informa al usuario

c) Eliminar miembro

Función: eliminar()

- Presenta listado de IDs activos
- Solicita ID a eliminar
- Busca su ubicación
- Si lo encuentra → cambia su estado (estado = -1)

- No reacomoda el árbol (se mantiene la estructura)

5.2 Recorridos del Árbol

Todas las funciones son recursivas (salvo recorrido por niveles).

a) Inorden

Función: inOrden(i)
Lógica: izquierda - raíz - derecha
Ideal para ver los IDs en orden ascendente.

b) Preorden

Función: preOrden(i)
Lógica: raíz - izquierda - derecha
Útil para observar la estructura jerárquica.

c) Postorden

Función: postOrden(i)
Lógica: izquierda - derecha - raíz
Se usa para procesar nodos hoja primero.

d) Por niveles

Función: porNiveles()

Recorre el puntero secuencialmente.
Muestra línea por línea según profundidad.
Solo se imprime un nodo cuando está activo.

5.3 Información del Árbol

1. Altura del árbol

Función: altura(i)
Calcula la mayor distancia desde un nodo hasta una hoja.
Si el árbol está vacío, la altura es 0.

2. Nodo raíz

Se accede a la posición 1 del puntero.
Si estado == 1, la raíz existe.

3. Nodos hoja

Función: nodosHoja(i)
Identifica nodos sin hijos activos.

4. Nodos internos

Función: nodosInternos(i)
Detecta nodos con al menos un hijo activo.

5. Peso del árbol

Función: peso()
Cuenta todos los nodos con estado activo.

5.4 Operaciones Avanzadas

1. Camino desde la raíz a un nodo

Función: camino()

- Solicita un ID
- Muestra la secuencia de IDs desde la raíz hasta el nodo encontrado

2. Longitud del camino

Función: longitudCamino(i, ID)
Devuelve cuántas aristas se recorren desde la raíz al nodo.

3. Altura de un nodo específico

Función: alturaNodo(i)
El usuario ingresa el ID, y se calcula la altura desde ese nodo hacia abajo.

4. Nivel de un nodo

Función: nivelNodo(i, ID)
Determina cuántos niveles por debajo de la raíz se encuentra el nodo.

5. Balanceo del árbol

En esta versión no está implementado en el código, pero se muestra en el menú.
Por lo tanto, la opción aparece como reservada para futuras versiones.

Nota importante: Para realizar balanceo del árbol, se debe utilizar el módulo complementario en Python descrito en la Parte II de este manual.

6. VALIDACIONES INCLUIDAS EN EL CÓDIGO C++

El programa incorpora restricciones clave:

- ID > 0
- Año > 0
- Género pertenece a {M, F}
- Un ID no puede duplicarse

- No se puede insertar cuando una posición está ocupada
- El nombre y rol no pueden ser cadenas vacías
- Eliminación solo marca como inactivo (no borra datos)

7. LIMITACIONES DEL SISTEMA C++

- El puntero tiene un máximo de 100 nodos
- Las búsquedas no son binarias; se realizan por recorrido secuencial
- Los nodos eliminados dejan "huecos" que no se reutilizan
- No hay rotaciones automáticas ni balanceo real
- La profundidad del árbol puede crecer mucho si los IDs se ingresan ordenados
- El menú muestra opciones no implementadas (balanceo)
- No incluye visualización gráfica del árbol

8. RECOMENDACIONES PARA USO EFICIENTE DEL SISTEMA C++

- Ingresar IDs variados para evitar un árbol desbalanceado
- Revisar la lista de IDs antes de buscar o eliminar
- No intentar insertar más de 100 miembros
- Utilizar los recorridos para verificar la estructura
- No reutilizar IDs eliminados
- Para visualizar gráficamente el árbol, utilizar el módulo Python complementario

9. RESOLUCIÓN DE PROBLEMAS FRECUENTES EN C++

Problema	Causa	Solución
No permite insertar	ID repetido o índice fuera de rango	Verificar listado de IDs previos
El nodo no aparece en recorridos	Estado = -1	Insertarlo con otro ID
Árbol muy profundo	Inserciones ordenadas	Variar los IDs al registrar
Camino o nivel incorrecto	El ID no existe	Revisar listado antes

10. FINALIZACIÓN DEL PROGRAMA C++

El sistema finaliza únicamente cuando el usuario selecciona la opción 5. Salir, cerrando todas las operaciones activas.

PARTE II: MÓDULO COMPLEMENTARIO EN PYTHON

11. OBJETIVO DEL MÓDULO PYTHON

Este módulo tiene como finalidad:

- Visualizar gráficamente la estructura del árbol binario de búsqueda mediante diagramas interactivos
- Balancear automáticamente el árbol para minimizar su altura y optimizar las operaciones de búsqueda
- Proporcionar todos los tipos de recorridos del árbol: inorden, preorden, postorden y por niveles
- Servir como herramienta educativa para entender visualmente cómo funcionan los árboles binarios
- Complementar el sistema principal en C++ con capacidades gráficas avanzadas

12. REQUISITOS DEL SISTEMA PYTHON

12.1 Software Necesario

- Python 3.7 o superior
- Librería Matplotlib (para visualización gráfica)

12.2 Instalación de Dependencias

Para instalar Matplotlib, ejecutar en la terminal:

```bash
pip install matplotlib
```

Nota: Matplotlib es la única librería externa requerida. Todo lo demás utiliza funciones estándar de Python.

13. ESTRUCTURA DEL PROGRAMA PYTHON

13.1 Clase Nodo

Representa cada elemento del árbol con los siguientes atributos:

- valor (int): Número almacenado en el nodo
- izquierda (Nodo*): Puntero al hijo izquierdo
- derecha (Nodo*): Puntero al hijo derecho

13.2 Clase ArbolBinario

Gestiona todas las operaciones del árbol:

- raiz (Nodo*): Puntero al nodo raíz del árbol

- Métodos de inserción, recorridos, balanceo y visualización

14. FUNCIONALIDADES DEL SISTEMA PYTHON

14.1 MENÚ PRINCIPAL

Al ejecutar el programa, se despliega el siguiente menú interactivo:

```
========== MENÚ ==========
1. Ingresar números
2. Dibujar árbol
3. Mostrar recorridos
4. Balancear árbol
5. Salir
==========================
```

14.2 OPCIÓN 1: INGRESAR NÚMEROS

Descripción: Permite insertar uno o varios valores numéricos al árbol.

Proceso:

1. El sistema solicita: "Ingresa números separados por espacio:"
2. El usuario ingresa valores (ejemplo: 50 30 70 20 40 60 80)
3. El sistema inserta cada valor siguiendo las reglas del ABB:
   - Valores menores van a la izquierda
   - Valores mayores van a la derecha
   - Valores duplicados se ignoran automáticamente

Validaciones:

- Solo acepta números enteros
- Valores no numéricos se omiten con mensaje de advertencia
- No hay límite en la cantidad de valores a ingresar

Ejemplo de uso:

```
Ingresa números separados por espacio: 10 5 15 3 7 12 20
Números insertados correctamente
```

14.3 OPCIÓN 2: DIBUJAR ÁRBOL

Descripción: Genera una representación gráfica visual del árbol utilizando Matplotlib.

Características de la visualización:

- Nodos: Círculos azul claro con borde azul oscuro
- Conexiones: Líneas negras que representan las relaciones padre-hijo
- Valores: Números centrados dentro de cada nodo
- Distribución automática: Los nodos se posicionan proporcionalmente según su nivel
- Información adicional: Muestra la altura actual del árbol en el título

Funcionamiento:

1. Calcula automáticamente las posiciones de cada nodo
2. Distribuye horizontalmente según el nivel para evitar superposiciones
3. Abre una ventana emergente con el gráfico interactivo
4. Permite zoom, desplazamiento y guardado de imagen

Mensaje de confirmación:

> Árbol visualizado en ventana gráfica

Nota importante: Si el árbol está vacío, muestra el mensaje: "El árbol está vacío"

14.4 OPCIÓN 3: MOSTRAR RECORRIDOS

Descripción: Ejecuta y muestra los cuatro tipos de recorridos del árbol.

a) Recorrido Inorden

- Orden: Izquierda → Raíz → Derecha
- Resultado: Valores en orden ascendente
- Uso: Verificar que el ABB esté correctamente estructurado

b) Recorrido Preorden

- Orden: Raíz → Izquierda → Derecha
- Resultado: Muestra la jerarquía del árbol
- Uso: Copiar o serializar la estructura del árbol

c) Recorrido Postorden

- Orden: Izquierda → Derecha → Raíz
- Resultado: Procesa hojas antes que nodos internos
- Uso: Eliminación segura de nodos o evaluación de expresiones

d) Recorrido Por Niveles

- Orden: Nivel por nivel, de izquierda a derecha
- Resultado: BFS (Búsqueda en Amplitud)
- Uso: Análisis por profundidad

Ejemplo de salida:

```
--- RECORRIDOS DEL ÁRBOL ---
Preorden:    [50, 30, 20, 40, 70, 60, 80]
Inorden:     [20, 30, 40, 50, 60, 70, 80]
Postorden:   [20, 40, 30, 60, 80, 70, 50]
Por niveles: [50, 30, 70, 20, 40, 60, 80]
```

14.5 OPCIÓN 4: BALANCEAR ÁRBOL

Descripción: Esta es la funcionalidad más importante del módulo. Reorganiza completamente el árbol para minimizar su altura y optimizar el rendimiento.

¿Qué es el balanceo?

El balanceo convierte un árbol desbalanceado (donde algunos caminos son mucho más largos que otros) en un árbol perfectamente equilibrado, donde la diferencia de altura entre subárboles es mínima.

Algoritmo de balanceo:

1. Extracción: Realiza un recorrido inorden para obtener todos los valores ordenados
2. Reconstrucción: Crea un nuevo árbol tomando siempre el elemento central como raíz
3. Recursión: Aplica el mismo proceso a los subárboles izquierdo y derecho

Ventajas del balanceo:

- Reduce la altura del árbol al mínimo posible
- Mejora drásticamente el tiempo de búsqueda
- Distribuye uniformemente los nodos
- Convierte un árbol degenerado (línea recta) en un árbol óptimo

Ejemplo práctico:

Antes del balanceo (árbol degenerado):

```
Valores insertados: 1 2 3 4 5 6 7
Estructura resultante:
1
 \
  2
   \
    3
     \
      4
       \
        5
         \
          6
           \
            7
Altura: 7
```

Después del balanceo:

```
Estructura resultante:
      4
     / \
    2   6
   / \ / \
  1  3 5  7
Altura: 3
```

Mensaje de confirmación:

```
Árbol balanceado correctamente (7 nodos)
```

Nota: Si el árbol está vacío, muestra: "El árbol está vacío, no hay nada que balancear"

¿Cuándo usar el balanceo?

- Después de insertar muchos valores en orden ascendente o descendente
- Cuando las búsquedas se vuelven lentas
- Para optimizar el árbol antes de realizar múltiples operaciones
- Como demostración educativa de optimización de estructuras de datos

14.6 OPCIÓN 5: SALIR

Finaliza la ejecución del programa de manera ordenada.

```
Hasta luego
```

15. FLUJO DE TRABAJO RECOMENDADO

Caso 1: Visualización básica

1. Ingresar números (Opción 1)
2. Dibujar árbol (Opción 2)
3. Ver recorridos (Opción 3)

Caso 2: Demostración de balanceo

1. Ingresar números ordenados: 1 2 3 4 5 6 7 (Opción 1)
2. Dibujar árbol desbalanceado (Opción 2) → Observar línea recta
3. Balancear árbol (Opción 4)
4. Dibujar árbol balanceado (Opción 2) → Observar árbol perfectamente distribuido
5. Comparar recorridos (Opción 3) → El inorden será idéntico

Caso 3: Análisis educativo

1. Ingresar valores aleatorios (Opción 1)
2. Ver altura actual con Dibujar (Opción 2)
3. Mostrar todos los recorridos (Opción 3)
4. Balancear y observar mejora (Opción 4)
5. Redibujar para ver el resultado (Opción 2)

16. VALIDACIONES Y MANEJO DE ERRORES EN PYTHON

El sistema incluye las siguientes validaciones:

Situación	Validación	Mensaje
Entrada no numérica	Se omite el valor	"'abc' no es un número válido, se omite."
Árbol vacío al dibujar	Verifica antes de graficar	"El árbol está vacío"
Árbol vacío al balancear	Verifica antes de operar	"El árbol está vacío, no hay nada que balancear"
Opción inválida	Valida entrada del menú	"Opción no válida, intenta de nuevo."

17. DIFERENCIAS ENTRE SISTEMA C++ Y MÓDULO PYTHON

Aspecto	Sistema C++	Módulo Python
Objetivo principal	Gestión completa del árbol genealógico	Visualización y balanceo

Aspecto	Sistema C++	Módulo Python
Tipo de datos	Estructura compleja (ID, nombre, año, género, rol)	Solo valores numéricos
Capacidad	Hasta 100 nodos (arreglo estático)	Ilimitada (memoria dinámica)
Visualización	No incluida	Gráficos interactivos con Matplotlib
Balanceo	No implementado	Totalmente funcional
Eliminación	Lógica (marca como inactivo)	No incluida
Complejidad	Sistema completo de gestión	Herramienta de análisis visual

18. INFORMACIÓN TÉCNICA

18.1 Complejidad de operaciones

Operación	Complejidad	Observaciones
Inserción	O(log n) promedio, O(n) peor caso	Depende del orden de inserción
Búsqueda	O(log n) promedio, O(n) peor caso	Mejora tras balanceo
Recorridos	O(n)	Visita todos los nodos
Balanceo	O(n)	Reconstruye el árbol completo
Visualización	O(n)	Calcula posición de cada nodo

18.2 Cálculo de altura

La altura se calcula recursivamente:

- Árbol vacío: altura = 0
- Nodo hoja: altura = 1
- Nodo interno: altura = 1 + max(altura_izq, altura_derecha)

18.3 Formato de almacenamiento

Los valores se almacenan en memoria mediante punteros dinámicos, sin límite de capacidad (excepto la memoria disponible del sistema).

19. CASOS DE USO EDUCATIVO

Demostración 1: Árbol desbalanceado vs balanceado

```
Insertar: 1 2 3 4 5 6 7 8 9 10
Altura antes de balancear: 10
Altura después de balancear: 4
Mejora: 60% de reducción
```

Demostración 2: Verificación de recorrido inorden

```
El recorrido inorden SIEMPRE da los valores ordenados
Esto confirma que el ABB está correctamente estructurado
```

Demostración 3: Análisis de complejidad

```
Árbol desbalanceado: búsqueda = O(n)
Árbol balanceado: búsqueda = O(log n)
```

20. RESOLUCIÓN DE PROBLEMAS EN PYTHON

Problema	Causa probable	Solución
No se muestra la ventana gráfica	Matplotlib no instalado	Ejecutar: pip install matplotlib
Error al balancear	Árbol vacío	Insertar valores primero (Opción 1)
Valores no se insertan	Entrada no numérica	Verificar formato: números separados por espacio
Ventana gráfica se cierra inmediatamente	Configuración de backend	Cerrar manualmente tras visualizar
"RecursionError" al dibujar	Bug corregido en versión actual	Usar la versión actualizada del código

21. RECOMENDACIONES DE USO DEL MÓDULO PYTHON

1. Siempre insertar valores antes de visualizar o balancear
2. Usar valores variados para observar mejor la estructura del árbol
3. Comparar la visualización antes y después del balanceo para entender su impacto
4. Experimentar con diferentes secuencias de inserción (ordenada, aleatoria, descendente)
5. Verificar el recorrido inorden para confirmar que el árbol está bien formado
6. Anotar la altura antes y después del balanceo para cuantificar la mejora

---

PARTE III: INTEGRACIÓN DE AMBOS SISTEMAS

22. INTEGRACIÓN ENTRE C++ Y PYTHON

Aunque ambos módulos operan de forma independiente, se complementan de las siguientes maneras:

22.1 Flujo de trabajo integrado:

1. En C++: Gestionar el árbol genealógico completo con todos los datos

2. En Python: Exportar solo los IDs para visualizar la estructura
3. En Python: Analizar el balance y altura del árbol
4. En C++: Aplicar insights del análisis visual

22.2 Ejemplo de datos exportables:

```
Sistema C++ tiene IDs: 150, 75, 200, 50, 100, 175, 250
↓
Exportar a Python: 150 75 200 50 100 175 250
↓
Visualizar y analizar estructura
↓
Identificar desbalanceo
↓
Aplicar estrategia de rebalanceo en C++ (manual)
```

23. LIMITACIONES DEL MÓDULO PYTHON

- Solo maneja valores numéricos enteros (no datos complejos como en C++)
- No persiste datos entre ejecuciones (todo en memoria)
- No incluye funcionalidad de eliminación de nodos
- Balanceo completo solo (no balanceo incremental)
- Requiere interfaz gráfica para visualización (no funciona en servidores sin GUI)

24. COMPARACIÓN FUNCIONAL COMPLETA

Funcionalidades exclusivas del sistema C++:

- Gestión completa de datos genealógicos
- Estructura de nodo compleja con múltiples campos
- Eliminación lógica de nodos
- Operaciones avanzadas (camino, longitud de camino, nivel de nodo)
- Persistencia en arreglo estático

Funcionalidades exclusivas del módulo Python:

- Visualización gráfica interactiva
- Balanceo automático del árbol
- Capacidad ilimitada de nodos
- Interfaz gráfica con Matplotlib

Funcionalidades compartidas:

- Inserción de nodos
- Recorridos: inorden, preorden, postorden, por niveles
- Cálculo de altura del árbol
- Búsqueda en el árbol

25. RECOMENDACIONES GENERALES

Para uso académico:

- Utilizar el sistema C++ para comprender la gestión completa de datos estructurados
- Utilizar el módulo Python para visualizar y comprender el concepto de balanceo
- Comparar rendimientos antes y después del balanceo

Para proyectos reales:

- Implementar el sistema C++ como base de datos principal
- Utilizar el módulo Python como herramienta de análisis y debugging
- Exportar IDs del sistema C++ al módulo Python para análisis visual

Para enseñanza:

- Demostrar primero la inserción y recorridos en C++
- Mostrar la visualización gráfica en Python
- Explicar el concepto de balanceo con ejemplos prácticos en Python
- Relacionar ambos sistemas para mostrar la integración de tecnologías

26. CONCLUSIÓN

Este sistema completo proporciona una solución integral para la gestión y análisis de árboles binarios de búsqueda:

Sistema C++:

- Sistema robusto de gestión de datos genealógicos
- Implementación completa de árbol binario de búsqueda
- Múltiples operaciones de análisis y consulta

Módulo Python:

- Capacidades visuales avanzadas
- Optimización automática mediante balanceo

- Herramienta educativa de alto valor

Uso recomendado: Utilizar ambos sistemas de manera complementaria para obtener una experiencia completa de gestión, visualización y optimización de estructuras de árbol binario de búsqueda, especialmente útil para propósitos educativos, desarrollo de software y análisis de datos.

27. CONTACTO Y SOPORTE

Para preguntas sobre el funcionamiento de cualquiera de los dos sistemas o su integración, consultar la documentación técnica completa o realizar pruebas con los casos de uso proporcionados en este manual.

Versión del documento: 1.0
Última actualización: 2025
Compatibilidad: C++ (compilador estándar), Python 3.7+, Matplotlib 3.0+